SANDIA REPORT

SAND88–1431 • UC–32
Unlimited Release
Printed August 1988

ALGEBRA — A Program That Algebraically Manipulates the Output of a Finite Element Analysis (EXODUS Version)

Amy P. Gilkey

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

53

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

SAND88-1431
Unlimited Release
Printed August 1988

Distribution
UC-32

ALGEBRA – A Program That Algebraically Manipulates the Output of a Finite Element Analysis (EXODUS Version)

Amy P. Gilkey
Applied Mechanics Division III
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

The ALGEBRA program allows the user to manipulate data from a finite element analysis before it is plotted. The finite element output data is in the form of variable values (e.g., stress, strain, and velocity components) in an EXODUS database. The ALGEBRA program evaluates user-supplied functions of the data and writes the results to an output EXODUS database which can be read by plot programs.

ACKNOWLEDGEMENTS

The original version of ALGEBRA was written by Mary R. Sagartz and Johnny H. Biffle [1].

The version of ALGEBRA described in [2] manipulates a SEACO database [3].

Contents

1. Introduction

The ALGEBRA program allows the user to manipulate data from a finite element program before it is plotted. The program reads the database output from an analysis program, manipulates the data using algebraic expressions supplied by the user, and writes the new data to a database to be processed by a plot program such as BLOT [4].

The program's algebraic evaluations allow special functions that are not provided by the analysis program (such as principal values, effective stress, and pressure) to be available for plotting. The evaluations include all of the FORTRAN arithmetic operations and most of the FORTRAN functions plus several special functions.

Both the input and output databases are in the EXODUS database format [6]. The EXODUS format defines four types of variables:

- A history variable has a value representative of the system as a whole at each time step (e.g., the total energy).
- A global variable is the same as a history variable except that global variables are only included in "whole" time steps (explained below).
- A nodal variable has a value for every node of the mesh at each whole time step in the analysis (e.g., the displacement in the x-direction).
- An element variable has a value for every element of the finite element analysis at each whole time step (e.g., the stress in the x-direction).

There are two types of time steps in an EXODUS database: a "history-only" time step contains the values for the history variables only; a "whole" time step includes the values for all the variables (history, global, nodal, and element).

Each element in the database is assigned to an "element block". An element block distinguishes a material or an element type (such as a truss or quadrilateral). A specific element variable may be undefined for some element blocks, meaning that the variable has no value for elements in that element block.

The algebraic expressions to be evaluated in ALGEBRA depend on the values from the input database. These input values include the time of the time step, the nodal coordinates, and the history, global, nodal, and element variables calculated by the analysis program, including values at specific nodes or elements. The values of variables from the previous database time step or the first database time step may also be

referenced in the algebraic expressions. History, global, nodal, and element variables are created by ALGEBRA in the output database, with the variable type determined by the types of variables in the expression being evaluated.

The EXODUS database format includes the names of the coordinates and variables. This allows the user to reference the input variables by name and to associate a meaningful name with calculated data.

There are two or three (depending on the number of dimensions in the mesh) special nodal variables which contain the displacement components at each node. The BLOT plot program [4] expects these variables to follow certain order and naming conventions. These variables must be the first nodal variables and they must start with "D" and end with the last letter of the corresponding coordinate name.

ALGEBRA allows the user to restrict the information that is written to the output database. The time steps to be written may be selected from those available on the input database. The size of the output mesh may be limited by giving the nodal coordinates of a section of the mesh or by selecting elements by element block.

2. Equation Input

The expressions to be evaluated are entered by the user as equations. The syntax is very similar to FORTRAN equation syntax. The first item is the variable to be assigned, followed by an "=", then the expression to be evaluated. The expression consists of constants, variables, arithmetic operators, and functions.

The equations must adhere to the following syntax rules.

- Blanks are treated as delimiters, but are otherwise ignored.
- Either lowercase or uppercase letters are acceptable, but lowercase letters are converted to uppercase.
- A "'" character in any equation starts a comment. The "'" and any characters following it on the line are ignored.
- An equation may be continued over several lines with a ">" character. The ">" and any characters following it on the current line are ignored and the next line is appended to the current line.

2.1 The Assigned Variable

The assigned variable name must start with a letter and can be up to eight alphanumeric characters (A–Z, 0–9) long. A name that is longer than eight characters is truncated with a warning. Blanks cannot be embedded in a variable name.

All assigned variables (except temporary variables specified by a DELETE command) will be written to the output database. The input database variables are not written to the output database unless they are assigned in an equation (such as X = X) or transferred with a SAVE command.

The type of the assigned variable depends on the expression. There are four types of "quantities" in an expression that are related to the variable types:

- History quantities include history variables, constants, and the time step time.
- Global quantities include global variables and nodal or element variables for specific nodes or elements.
- Nodal quantities include nodal variables and nodal coordinates, unless the value is limited to a specific node.

- Element quantities include element variables, unless the value is limited to a specific element.

History and global quantities are referred to as "single-value" quantities. Nodal and element quantities are referred to as "arrays".

Each part of an expression yields a result of a particular type. The types of constants and variables are defined above. The type of an arithmetic operation is dependent on the types of its operands. If both operands are history quantities, the operation yields a history quantity. If the operands are global and/or history quantities, a global quantity results. If either operand is an array, the operation type is the array type. Thus a nodal quantity and an element quantity cannot appear in the same operation. For array operations, the operator is applied to each array element. The type of a function is dependent on the types of its parameters. The rules for operand types also apply to all function parameters. One special type of function yields a global quantity regardless of the parameter type.

Equations that result in a history variable are evaluated for each time step. Equations resulting in other types of variables are evaluated only for "whole" time steps.

The assigned variable can be reassigned, but it must be assigned to the same variable type (history, global, nodal or element).

The equations are evaluated in order. The assigned variables are grouped by variable type, but are otherwise output in the order they are first assigned by the equations.

2.2 Restricting the Nodes and/or Elements

Nodes and/or elements may be deleted from the input database with the ZOOM or VISIBLE commands. An input variable is defined for all input nodes and/or elements. An output variable is only defined for the nodes and/or elements to be output.

Element variables may be undefined for certain element blocks. This may be further restricted with the BLOCKS command. If two or more element variables are combined with an operator or are function parameters, the resulting variable is only defined for an element block if all the variables involved are defined for that block.

When an operation or function is performed on an array variable, it is only performed for the defined nodes/elements. This is done to prevent problems with numerical errors such as divide by zero for undefined values.

2.3 Constants

Constants can be entered in any legal FORTRAN numeric format (e.g., 5, 5.4 or 5.4E3). All integers are converted to real numbers. If the constant is signed, parenthesis should surround the sign and constant.

2.4 Variables

The variables that may be found in the expression to be evaluated are:

- any input database history, global, nodal or element variable,
- the values for any coordinate,
- a reference to a specific nodal or element quantity,
- the time associated with each time step, and
- any previously assigned variable.

If an embedded blank is included in an input database variable or coordinate name, the blank must be deleted in references to the variable. For example, variable "SIG X" must be entered as "SIGX".

The coordinates may be referenced in the expression by name. They are treated as an input database nodal variable whose value remains constant in all "whole" time steps.

If the value for a specific node or element is desired, a "$" and the node or element number is appended to the variable name. For example, SIGR$40 refers to the value for the 40^{th} element of variable SIGR. A specifier may be appended to the name of any nodal or element quantity in an expression, including coordinates and previously assigned variables. References to specific nodes and/or elements can only be made if the variable is defined at that node and/or element.

The value of a variable in the previous time step is referenced by appending a ":" to the variable name. The value in the first time step is referenced by appending a ":1" to the variable name. If time steps are selected, the previous and first time steps refer to the selected time steps, not the input time steps.

The name "TIME" is reserved for the time associated with each time step. The output database times are copied from the input database unless a value is assigned to the variable TIME. The TIME expression must evaluate to a history quantity. TIME may also appear in the expression, referring to the input or assigned database time.

The equations are evaluated in order. References to a variable name in the expression refer to the last assigned value, or to the input variable if the name has not been

assigned. For example, if input global variable CONST has a value of 4 and the following equations are executed,

```
X = CONST
CONST = 2*CONST
Y = CONST
```

the result is X equals 4, CONST equals 8, and Y equals 8.

2.5 Operators

The legal operations are addition (+), subtraction (−), multiplication (*), division (/), and exponentiation (**). The operands may be either single-value or array quantities as explained in Section 2.1.

FORTRAN operator precedence rules apply (e.g., multiplication is performed before addition). Parenthesis may be used to change the order of evaluation.

Two operators cannot be placed in succession. To precede a value with a sign, enclose the sign and value in parenthesis. For example,

```
A = -5 * -SIN(0.5)
```

should be written as

```
A = (-5) * (-SIN(0.5))
```

where the parenthesis around the −5 are optional.

2.6 Functions

Many of the standard FORTRAN functions and several special functions are implemented in ALGEBRA. These functions are summarized in Appendix B. The parameters for any function may be expressions and all parameters must be supplied. The parameters may be either single-value or array quantities as explained in Section 2.1.

A function in an equation is distinguished from a variable name by the "(" which follows the function name. This allows the user to assign variable names which are the same as the function names and to reference input database variables with the same names as the functions.

FORTRAN Functions

The standard FORTRAN functions implemented are: AINT, ANINT, ABS, MOD, SIGN, DIM, MAX, MIN, SQRT, EXP, LOG, LOG10, SIN, COS, TAN, ASIN, ACOS, ATAN, ATAN2, SINH, COSH, and TANH. The use and result of these functions is the same as in FORTRAN, and the same restrictions apply.

Tensor Principal Values and Magnitude Functions

Functions PMAX and PMIN calculate the maximum and minimum principal values of a symmetric tensor. For example, to obtain the maximum principal values for a tensor T,

$$\mathsf{SMAX} = \mathsf{PMAX}\ (T_{11},\ T_{22},\ T_{33},\ T_{12},\ T_{23},\ T_{31}).$$

For a two-dimensional tensor or a tensor using cylindrical coordinates for an axisymmetric solution, PMAX2 and PMIN2 may be used:

$$\mathsf{SMAX} = \mathsf{PMAX2}\ (T_{11},\ T_{22},\ T_{12}).$$

The function TMAG calculates the magnitude of the deviatoric part of a symmetric tensor. To calculate the magnitude of tensor T,

$$\mathsf{SMAG} = \mathsf{TMAG}\ (T_{11},\ T_{22},\ T_{33},\ T_{12},\ T_{23},\ T_{31})$$

where the following calculation is made:

$$\mathsf{SMAG} = \sqrt{(T_{11}-T_{22})^2 + (T_{22}-T_{33})^2 + (T_{33}-T_{11})^2 + 6*(T_{12}^2+T_{23}^2+T_{31}^2)}.$$

To obtain the von Mises stress, the value supplied by function TMAG is multiplied by the constant $1/\sqrt{2}$. To calculate effective strain, multiply by the constant $\sqrt{2.0}/3.0$.

IF Functions

The functions IFLZ, IFEZ, and IFGZ provide a simple if-then-else capability. Each function expects three parameters: a condition, a true result, and a false result. Function IFLZ returns the true result if the condition evaluates to less than zero; otherwise the function returns the false result. Function IFEZ checks for equal to zero and IFGZ checks for greater than zero. For example, the equation

$$x = \mathsf{IFLZ}\ (\mathit{cond},\ \mathit{rtrue},\ \mathit{rfalse})$$

with global parameters *cond*, *rtrue*, and *rfalse* could be implemented in FORTRAN by

```
IF (cond .LT. 0.0) THEN
      x = rtrue
ELSE
      x = rfalse
END IF
```

All the parameters are evaluated before the function, so both the true result and the false result are evaluated even though only one is needed.

Array ⇒ Global Variable Functions

The functions SUM, SMAX, and SMIN perform a calculation on a nodal or element array parameter which produces a global result. SUM sums all the array values. SMAX and SMIN return the maximum and minimum of all the array values.

Values for specific nodes and/or elements are only included in the function calculation if the variable is defined at that node and/or element.

Envelope Functions

An "envelope" function performs a calculation that is cumulative for all previous time steps. The function ENVMAX results in an array (assuming the parameter is an array) that is the maximum of each array value for all previous selected time steps and the current time step. On the last time step, ENVMAX contains the maximum of each array value for all selected time steps. ENVMIN is the corresponding minimum function.

3. Command Input

The user can issue a command whenever an equation is expected. The commands are in free-format and must adhere to the following syntax rules.

- Valid delimiters are a comma or one or more blanks.
- Either lowercase or uppercase letters are acceptable, but lowercase letters are converted to uppercase.
- A "!" character in any command line starts a comment. The "!" and any characters following it on the line are ignored.
- A command may be continued over several lines with an ">" character. The ">" and any characters following it on the current line are ignored and the next line is appended to the current line.

Each command has an action keyword or "verb" followed by a variable number of parameters.

The command verb is a character string. It may be abbreviated, as long as enough characters are given to distinguish it from other commands.

The meaning and type of the parameters is dependent on the command verb. Most command parameters are optional. If an optional parameter field is blank, a command-dependent default value is supplied. Below is a description of the valid entries for parameters.

- A numeric parameter may be a real number or an integer. A real number may be in any legal FORTRAN numeric format (e.g., 1, 0.2, -1E-2). An integer parameter may be in any legal integer format.
- A string parameter is a literal character string. Most string parameters may be abbreviated.
- Variable names must be fully specified. The blank delimiter creates a problem with database variable names with embedded blanks. The program handles this by deleting all embedded blanks from the input database names. For example, the variable name "SIG R" must be entered as "SIGR". The blank must be deleted in any references to the variable. All database names appear in uppercase without the embedded blanks in all displays.

The notation conventions used in the command descriptions are:

- The command verb is in **bold** type.
- A literal string is in all uppercase SANSERIF type and should be entered as shown (or abbreviated).
- The value of a parameter is represented by the parameter name in *italics*.
- A literal string in square brackets ("[ ]") represents a parameter option which is omitted entirely (including any following comma) if not appropriate. These parameters are distinct from most parameters in that they do not require a comma as a place holder to request the default value.
- The default value of a parameter is in angle brackets ("< >"). The initial value of a parameter set by a command is usually the default parameter value. If not, the initial setting is given with the default or in the command description.

The commands are summarized in Appendix C.

3.1 Database Editing Commands

TITLE

TITLE sets the title to be written to the output database. The title is input on the next line. If no TITLE command is issued, the input database title is written to the output database.

3.2 Variable Selection Commands

SAVE *variable*$_1$, *variable*$_2$, ... or *option*$_1$, *option*$_2$, ... <no default>

SAVE transfers variables from the input database to the output database. An individual variable may be transferred by listing its name as a parameter. For example,

SAVE Y, Z

has the same effect as the equations (with the exception noted below):

Y = Y
Z = Z.

Assigned variables are affected by the BLOCKS command; SAVEd variables are not.

The following *options* transfer sets of variables:

SAVE HISTORY

transfers all input database history variables.

SAVE GLOBAL

transfers all input database global variables.

SAVE NODAL

transfers all input database nodal variables.

SAVE ELEMENT

transfers all input database element variables.

SAVE ALL

transfers all input database history, global, nodal, and element variables.

The SAVE command causes the variables to be output in the same order they would be if they were assigned by equations at that point.

If a SAVEd variable is also an assigned variable, the assigned value is written to the output database, regardless of whether the SAVE is done before or after the assignment.

DELETE *variable*$_1$, *variable*$_2$, ... <no default>

DELETE marks an assigned variable as a temporary variable that will not be written to the output database. A variable must be assigned (or SAVEd) before it is listed in a DELETE command.

3.3 Time Step Selection Commands

ALGEBRA allows the user to restrict the time steps from the input database that are written to the output database. By default, all the time steps from the input database are written to the output database.

Time step selection is performed in one of the following modes:

- Interval-Times Mode selects time steps at uniform intervals between a minimum and a maximum time. If this mode has a delta offset, the first selected time is not the minimum time, but the minimum time plus the interval. If this mode has a zero offset, the first selected time is the minimum time.

- All-Available-Times Mode selects all time steps between a minimum and a maximum time.

- User-Selected-Times Mode selects time steps which are explicitly specified by the user.

The nearest time step from the database is chosen for each selected time.

The following are the time step selection parameters:

- *tmin* is the minimum selected time,
- *tmax* is the maximum selected time,
- *nintv* is the number of selected time intervals, and
- *delt* is the selected time interval.

In the interval-times mode, up to *nintv* time steps at interval *delt* between *tmin* and *tmax* are selected. The mode may have a delta offset or a zero offset. With a delta offset, the first selected time is *tmin*+*delt*; with a zero offset, it is *tmin*.

In the interval-times mode with a delta offset, the number of selected time intervals *nintv* and the selected time interval *delt* are related mathematically by the equations:

$$delt = (tmax - tmin)/nintv \quad (1)$$
$$nintv = \text{int}\ ((tmin - tmax)/delt) \quad (2)$$

With a zero offset, *nintv* and *delt* are related mathematically by the equations:

$$delt = (tmax - tmin)/(nintv - 1) \quad (1)$$
$$nintv = \text{int}\ ((tmin - tmax)/delt) + 1 \quad (2)$$

The user specifies either *nintv* or *delt*. If *nintv* is specified, *delt* is calculated using equation 1. If *delt* is specified, *nintv* is calculated using equation 2.

In the all-available-times mode, all database time steps between *tmin* and *tmax* are selected (parameters *nintv* and *delt* are ignored). In the user-selected-times mode, the specified times are selected (all parameters are ignored).

TMIN *tmin* <minimum database time>

TMIN sets the minimum selected time *tmin* to the specified parameter value. If the user-selected-times mode is in effect, the mode is changed to the all-available-times mode.

In interval-times mode, if *nintv* is selected (by a NINTV or ZINTV command), *delt* is calculated. If *delt* is selected (by a DELTIME command), *nintv* is calculated.

TMAX *tmax* <maximum database time>

TMAX sets the maximum selected time *tmax* to the specified parameter value. If the user-selected-times mode is in effect, the mode is changed to the all-available-times mode.

In interval-times mode, if *nintv* is selected (by a NINTV or ZINTV command), *delt* is calculated. If *delt* is selected (by a DELTIME command), *nintv* is calculated.

NINTV *nintv* <10 or the number of database time steps − 1, whichever is smaller>

NINTV sets the number of selected time intervals *nintv* to the specified parameter value and changes the mode to the interval-times mode with a delta offset. The selected time interval *delt* is calculated.

ZINTV *nintv* <10 or the number of database time steps, whichever is smaller>

ZINTV sets the number of selected time intervals *nintv* to the specified parameter value and changes the mode to the interval-times mode with a zero offset. The selected time interval *delt* is calculated.

DELTIME *delt* <$(tmax - tmin)/(nintv - 1)$, where *nintv* is 10 or the number of database time steps, whichever is smaller>

DELTIME sets the selected time interval *delt* to the specified parameter value and changes the mode to the interval-times mode with a zero offset. The number of selected time intervals *nintv* is calculated.

ALLTIMES

ALLTIMES changes the mode to the all-available-times mode.

TIMES [ADD,] t_1, t_2, ... <no times selected>

TIMES changes the mode to the user-selected-times mode and selects times t_1, t_2, etc. The closest time step from the database is selected for each specified time.

Normally, a TIMES command selects only the listed time steps. If ADD is the first parameter, the listed steps are added to the current selected times. Any other time step selection command clears all TIMES selected times.

Up to the maximum number of time steps in the database may be specified. Times are selected in the order encountered on the database, regardless of the order the times are specified in the command. Duplicate references to a time step are ignored.

STEPS [ADD,] n_1, n_2, ... <no steps selected>

The STEPS command is equivalent to the TIMES command except that it selects time steps by the step number, not by the step time.

HISTORY ON or OFF <ON>

HISTORY controls whether history time steps are included in the selected time steps (if ON) or only whole time steps (if OFF).

For example, if the times from the database are 0.0, 0.5, 1.0, 1.5, etc., the commands

```
TMIN 0.0
TMAX 5.0
NINTV 5
```

select times 1.0, 2.0, 3.0, 4.0, and 5.0. If the NINTV command is replaced by

```
ZINTV 3
```

then times 0.0, 2.5, and 5.0 are selected. If the NINTV command is replaced by

```
DELTIME 2.0
```

then times 0.0, 2.0, 4.0 are selected.

Another example is given in Appendix D.

3.4 Mesh Limiting Commands

These commands limit the mesh that is written to the output database by deleting nodes and elements that do not satisfy the limiting conditions. A deleted node or element is entirely removed from the output database and is ignored in all equation evaluations. Deleting nodes and elements may cause the nodes and elements on the output database to be numbered differently than those on the input database.

If both the ZOOM and VISIBLE commands are in effect, the nodes and elements must satisfy both the limiting conditions to be written to the output database.

By default, the entire mesh is written to the output database.

ZOOM *xmin*, *xmax*, *ymin*, *ymax*, *zmin*, *zmax* <no default>

ZOOM sets the limits of the mesh to be written to the output database. Limits *xmin* to *xmax* apply to the first coordinate, *ymin* to *ymax* to the second coordinate, and *zmin* to *zmax* to the third coordinate (if any). A node is deleted if it is not within the rectangle (or brick) defined by these limits. An element is deleted if all of its nodes are deleted.

VISIBLE [ADD or DELETE,] $block\_id_1$, $block\_id_2$, ... <all element blocks>

VISIBLE limits the element blocks to be written to the output database. An element that is not in a "visible" element block is deleted. A node is deleted if all the elements containing the node are deleted.

The *block_id* refers to the element block identifier (displayed by the LIST BLOCKS command).

If there is no parameter, all element blocks are visible. If the first parameter is ADD or DELETE, the element blocks listed are added to or deleted from the current visible set. Otherwise, only the element blocks listed in the command are visible.

3.5 Element Block Selection Commands

BLOCKS [ADD or DELETE,] *block_id*$_1$, *block_id*$_2$, ... <all element blocks>

BLOCKS selects the element blocks which have defined values for all following equations. An element variable can be defined for an element block only if that block is selected. This command can only mark element variables as undefined, it cannot mark previously undefined variables as defined. It has no effect on nodal variables.

The BLOCKS command affects all following equations unless another BLOCKS command is entered. The BLOCKS command has no effect on the output of SAVEd element variables.

The *block_id* refers to the element block identifier (displayed by the LIST BLOCKS command).

If there is no parameter, all element blocks are selected. If the first parameter is ADD or DELETE, the element blocks listed are added to or deleted from the current selected set. Otherwise, only the element blocks listed in the command are selected.

MATERIAL [ADD or DELETE,] *block_id*$_1$, *block_id*$_2$, ... <all element blocks>

MATERIAL is exactly equivalent to a BLOCKS command.

3.6 Information and Termination Commands

SHOW *command* <no parameter>

SHOW displays the settings of parameters relevant to the *command.* For example, SHOW TMIN displays the time step selection criteria.

SHOW with no parameters displays any nondefault command parameters and all input equations.

LIST *option* <no parameter>

LIST displays database information, depending on the *option.*

LIST VARS

displays a summary of the database. The summary includes the database title; the number of nodes, elements, and element blocks; the number of node sets and side sets; and the number of variables.

LIST BLOCKS or MATERIAL

displays a summary of the element blocks. The summary includes the block identifier, the number of elements in the block, the number of nodes per element, and the number of attributes per element.

LIST QA

displays the QA records and the information records.

LIST NAMES

displays the names of the history, global, nodal, and element variables.

LIST STEPS

displays the number of time steps and the minimum and maximum time step times.

LIST TIMES

displays the step numbers and times for all time steps on the database.

HELP *option* <no parameter>

HELP displays information about the ALGEBRA program, depending on the *option.*

HELP RULES

displays a summary of the equation syntax rules.

HELP COMMANDS

displays a summary of the commands.

HELP FUNCTIONS

lists the names of all available functions and displays some useful equations, such as the equation for effective strain.

HELP

lists the available HELP options and displays any nondefault command parameters and all input equations.

LOG

LOG requests that the log file be saved when the program is exited. Each correct equation and command that the user enters (excluding informational commands such as SHOW) is written to the log file.

END

END ends the equation input and begins the equation evaluation. The word "EXIT" may be used in place of "END".

QUIT

QUIT ends the equation input and exits the program immediately without writing an output database.

4. The Output EXODUS Database

The EXODUS database format is briefly described in Appendix A. The first part of the EXODUS database consists of the mesh description in the GENESIS database format [5]. The mesh description includes the nodal coordinates, the element block information (including the element connectivity), the node sets, and the side sets. The second part of the database contains the time step information, including all the variable values for each time step.

If nodes and/or elements have been deleted from the database with a ZOOM or VISIBLE command, the entire output database reflects the deletions and any node or element renumbering caused by the deletions.

The output database mesh description is copied (with changes for deletions) from the input database. The database title may be changed with the TITLE command.

All QA records from the input database are copied to the output database, and a record is added describing the current ALGEBRA run. All input database informational records are copied to the output database.

All names on the output database are in uppercase and have all embedded blanks removed. The coordinate and element block names from the input database are converted and copied (with changes for deletions) to the output database. The output variable names are assigned in the equations.

The output database element variable truth table has an entry for each output element variable which indicates whether the variable is defined for each element block. This is determined by the input element variable truth table, the equations, and the BLOCKS command.

The output time steps include the time step times and the output variables for each time step. Each selected input time step is processed; non-selected time steps are ignored. For a history-only time step, only the history variables are evaluated and written out. For a whole time step, all variables are evaluated and written to the output database.

5. Informational and Error Messages

ALGEBRA operates in three stages: (1) it scans the input database for general information, (2) it inputs commands and equations from the user, (3) it re-reads the input database and copies the mesh description to the output database, and (4) it evaluates the equations for each time step.

ALGEBRA expects a valid database. If a format error is discovered before the time steps, the program prints an error of the following format:

DATABASE ERROR - Reading *database item*

and aborts. This problem may occur either while scanning the input database or while copying the mesh description to the output database.

If a format error is found while reading the time steps, the following error message is printed:

WARNING - End-of-file during time steps

or

DATABASE ERROR - Reading *database item.*

If this error is encountered while scanning the input database, the time step with the error and all following time steps are ignored, but the program continues and the previous time steps are available for processing. Some database errors may not be detected until the equations are being evaluated. The program aborts when the error is encountered, but the output database is correct for all previous time steps.

An equation is checked for syntax errors as soon as the user enters the line. If an error is found, a message is printed and the equation is ignored (with a message to that effect). If only a warning is printed, the equation is accepted. If the message is not sufficiently informative, the description of the equation syntax (Chapter 2) may be helpful.

A command is performed as soon as it is entered. A command error usually causes the command to be ignored. The command is usually performed if only a warning is printed. The display after the command shows the effect of the command. If the message is not sufficiently informative, the appropriate command description (Chapter 3) may be helpful.

The evaluation loop processes each time step by reading the needed input database variables, evaluating the equations, and writing the results to the output database.

Any error during this stage causes the program to abort (with a fatal error message). The output database is readable, but it contains only the data from the time steps processed before the error.

A numerical error while evaluating the equations (such as divide by zero) causes a fatal error. A message is printed describing the error and the equation that caused the error is displayed after the error message.

The program allocates memory dynamically as it is needed. If the system runs out of memory, the following message is printed:

FATAL ERROR - Too much dynamic memory requested

and the program aborts. The user should first try to obtain more memory on the system. Another solution is to run the program in a less memory-intensive fashion. For example, entering fewer equations may require less memory.

ALGEBRA has certain programmer-defined limitations (for example, the number of curves that may be defined. The limits are not specified in this manual since they may change. In most cases the limits are chosen to be more than adequate. If the user exceeds a limit, a message is printed. If the user feels the limit is too restrictive, the code sponsor should be notified so the limit may be raised in future releases of ALGEBRA.

6. Executing ALGEBRA

The details of executing ALGEBRA are dependent on the system being used. The system manager of any system that runs ALGEBRA should provide a supplement to this manual that explains how to run the program on that particular system. Site supplements for all currently supported systems are in Appendix E.

6.1 Execution Files

The table below summarizes ALGEBRA's file usage.

Description	Unit Number	Type	File Format
User input	standard input	input	Section 2 and 3
User output	standard output	output	ASCII
EXODUS database	11	input	Appendix A
EXODUS database	12	output	Appendix A
Log file	99	optional output	ASCII

All files must be connected to the appropriate unit before ALGEBRA is run. Each file (except standard input and output) is opened with the name retrieved by the EXNAME routine of the SUPES library [8].

6.2 Special Software

ALGEBRA is written in ANSI FORTRAN-77 [7] with the exception of the following system-dependent features:

- the OPEN options for the files and
- the use of ASCII characters that are not in the FORTRAN standard character set.

ALGEBRA uses the following software packages:

- the SUPES package [8] which includes dynamic memory allocation, a free-field reader, and FORTRAN extensions and
- the SLATEC mathematics library to calculate the principle values of second-order tensors.

References

[1] Mary R. Sagartz and Johnny H. Biffle, "ALGEBRA – A Computer Program That Algebraically Manipulates Finite Element Output Data," SAND80-2061, Sandia National Laboratories, Albuquerque, NM, September 1980.

[2] Amy P. Gilkey, "ALGEBRA – A Program That Algebraically Manipulates the Output of a Finite Element Analysis," SAND86-0881, Sandia National Laboratories, Albuquerque, NM, May 1986.

[3] Zelma E. Beisinger, "SEACO: Sandia Engineering Analysis Department Code Output Data Base," SAND84-2004, Sandia National Laboratories, Albuquerque, NM, in preparation.

[4] Amy P. Gilkey, "BLOT – A Mesh and Curve Plot Program for the Output of a Finite Element Analysis," SAND88-1432, Sandia National Laboratories, Albuquerque, NM, in preparation.

[5] Lee M. Taylor, Dennis P. Flanagan, and William C. Curran, "The GENESIS Finite Element Mesh File Format," SAND86-0910, Sandia National Laboratories, Albuquerque, NM, May 1986.

[6] William C. Mills-Curran, Amy P. Gilkey, Dennis P. Flanagan, "EXODUS: A Finite Element File Format for Pre- and Post-Processing," SAND87-2997, Sandia National Laboratories, Albuquerque, NM, in preparation.

[7] *American National Standard Programming Language FORTRAN*, American National Standards Institute, ANSI X3.9-1978, New York, 1978.

[8] Dennis P. Flanagan, William C. Curran, and Lee M. Taylor, "SUPES – A Software Utility Package for Engineering Science," SAND86-0911, Sandia National Laboratories, Albuquerque, NM, September 1986.

A. The EXODUS Database Format

The following code segment reads an EXODUS database. The first segment is the GENESIS database format.

```
C   --Open the EXODUS database file

      NDB = 9
      OPEN (UNIT=NDB, ..., STATUS='OLD', FORM='UNFORMATTED')

C   --Read the title

      READ (NDB) TITLE
C       --TITLE - the title of the database (CHARACTER*80)

C   --Read the database sizing parameters

      READ (NDB) NUMNP, NDIM, NUMEL, NELBLK,
     &   NUMNPS, LNPSNL, NUMESS, LESSEL, LESSNL, NVERSN
C       --NUMNP - the number of nodes
C       --NDIM - the number of coordinates per node
C       --NUMEL - the number of elements
C       --NELBLK - the number of element blocks
C       --NUMNPS - the number of node sets
C       --LNPSNL - the length of the node sets node list
C       --NUMESS - the number of side sets
C       --LESSEL - the length of the side sets element list
C       --LESSNL - the length of the side sets node list
C       --NVERSN - the file format version number

C   --Read the nodal coordinates

      READ (NDB) ((CORD(INP,I), INP=1,NUMNP), I=1,NDIM)

C   --Read the element order map (each element must be listed once)

      READ (NDB) (MAPEL(IEL), IEL=1,NUMEL)
```

```
C   --Read the element blocks

      DO 100 IEB = 1, NELBLK

C       --Read the sizing parameters for this element block

          READ (NDB) IDELB, NUMELB, NUMLNK, NATRIB
C           --IDELB - the element block identification (must be unique)
C           --NUMELB - the number of elements in this block
C           --   (the sum of NUMELB for all blocks must equal NUMEL)
C           --NUMLNK - the number of nodes defining the connectivity
C           --   for an element in this block
C           --NATRIB - the number of element attributes for an element
C           --   in this block

C       --Read the connectivity for all elements in this block

          READ (NDB) ((LINK(J,I), J=1,NUMLNK, I=1,NUMELB)

C       --Read the attributes for all elements in this block

          READ (NDB) ((ATRIB(J,I), J=1,NATRIB, I=1,NUMELB)

  100 CONTINUE
```

```
C   --Read the node sets

      READ (NDB) (IDNPS(I), I=1,NUMNPS)
C      --IDNPS - the ID of each node set
      READ (NDB) (NNNPS(I), I=1,NUMNPS)
C      --NNNPS - the number of nodes in each node set
      READ (NDB) (IXNNPS(I), I=1,NUMNPS)
C      --IXNNPS - the index of the first node in each node set
C      --   (in LTNNPS and FACNPS)

      READ (NDB) (LTNNPS(I), I=1,LNPSNL)
C      --LTNNPS - the nodes in all the node sets
      READ (NDB) (FACNPS(I), I=1,LNPSNL)
C      --FACNPS - the factor for each node in LTNNPS

C   --Read the side sets

      READ (NDB) (IDESS(I), I=1,NUMESS)
C      --IDESS - the ID of each side set
      READ (NDB) (NEESS(I), I=1,NUMESS)
C      --NEESS - the number of elements in each side set
      READ (NDB) (NNESS(I), I=1,NUMESS)
C      --NNESS - the number of nodes in each side set
      READ (NDB) (IXEESS(I), I=1,NUMESS)
C      --IXEESS - the index of the first element in each side set
C      --   (in LTEESS)
      READ (NDB) (IXNESS(I), I=1,NUMESS)
C      --IXNESS - the index of the first node in each side set
C      --   (in LTNESS and FACESS)

      READ (NDB) (LTEESS(I), I=1,LESSEL)
C      --LTEESS - the elements in all the side sets
      READ (NDB) (LTNESS(I), I=1,LESSNL)
C      --LTNESS - the nodes in all the side sets
      READ (NDB) (FACESS(I), I=1,LESSNL)
C      --FACESS - the factor for each node in LTNESS
```

A valid GENESIS database may end at this point or at any point until the number of variables is read.

```
C   --Read the QA header information

      READ (NDB, END=900) NQAREC
C      --NQAREC - the number of QA records (must be at least 1)

      DO 110 IQA = 1, MAX(1,NQAREC)
         READ (NDB) (QATITL(I,IQA), I=1,4)
C         --QATITL - the QA title records; each record contains:
C         --   1) analysis code name (CHARACTER*8)
C         --   2) analysis code qa descriptor (CHARACTER*8)
C         --   3) analysis date (CHARACTER*8)
C         --   4) analysis time (CHARACTER*8)
  110 CONTINUE

C   --Read the optional header text

      READ (NDB, END=900) NINFO
C      --NINFO - the number of information records

      DO 120 I = 1, NINFO
         READ (NDB) INFO(I)
C         --INFO - extra information records (optional) that contain
C         --   any supportive documentation that the analysis code
C         --   developer wishes (CHARACTER*80)
  120 CONTINUE

C   --Read the coordinate names

      READ (NDB, END=900) (NAMECO(I), I=1,NDIM)
C      --NAMECO - the coordinate names (CHARACTER*8)

C   --Read the element type names

      READ (NDB, END=900) (NAMELB(I), I=1,NELBLK)
C      --NAMELB - the element type names (CHARACTER*8)
```

The GENESIS section of the database ends at this point.

```
C   --Read the history, global, nodal, and element variable information

      READ (NDB, END=900) NVARHI, NVARGL, NVARNP, NVAREL
C       --NVARHI - the number of history variables
C       --NVARGL - the number of global variables
C       --NVARNP - the number of nodal variables
C       --NVAREL - the number of element variables

      READ (NDB)
     &   (NAMEHV(I), I=1,NVARHI),
     &   (NAMEGV(I), I=1,NVARGL),
     &   (NAMENV(I), I=1,NVARNP),
     &   (NAMEEV(I), I=1,NVAREL)
C       --NAMEHI - the history variable names (CHARACTER*8)
C       --NAMEGV - the global variable names (CHARACTER*8)
C       --NAMENV - the nodal variable names (CHARACTER*8)
C       --NAMEEV - the element variable names (CHARACTER*8)

      READ (NDB) ((ISEVOK(I,J), I=1,NVAREL), J=1,NELBLK)
C       --ISEVOK - the name truth table for the element blocks;
C       --   ISEVOK(i,j) refers to variable i of element block j;
C       --   the value is 0 if and only if data will NOT be output for
C       --   variable i for element block j (otherwise the value is 1)
```

```
C   --Read the time steps

  130 CONTINUE
        READ (NDB, END=900) TIME, HISTFL
C        --TIME - the time step value
C        --HISTFL - the time step type flag:
C        --   0.0 for all variables output ("whole" time step) else
C        --   only history variables output ("history-only" time step)
C
        READ (NDB) (VALHV(IVAR), IVAR=1,NVARGL)
C        --VALHV - the history values for the current time step

        IF (HISTFL .EQ. 0.0) THEN
           READ (NDB) (VALGV(IVAR), IVAR=1,NVARGL)
C           --VALGV - the global values for the current time step

           DO 140 IVAR = 1, NVARNP
              READ (NDB) (VALNV(INP,IVAR), INP=1,NUMNP)
C              --VALNV - the nodal variables at each node
C              --   for the current time step
  140         CONTINUE

           DO 160 IBLK = 1, NELBLK
              DO 150 IVAR = 1, NVAREL
                 IF (ISEVOK(IVAR,IBLK) .NE. 0) THEN
                    READ (NDB) (VALEV(IEL,IVAR,IBLK),
     &                   IEL=1,NUMELB(IBLK))
C                    --VALEV - the element variables at each element
C                    --   for the current time step
                 END IF
  150         CONTINUE
  160       CONTINUE
        END IF

C      --Handle time step data
         ...
      GOTO 130

  900 CONTINUE
C   --Handle end of file on database
      ...
```

B. Summary of Functions

Standard FORTRAN Functions	
r = AINT (x)	truncation: $\lvert x \rvert$
r = ANINT (x)	nearest integer: $[x + .5*\text{sign}(x)]$
r = ABS (x)	absolute value: $\lvert x \rvert$
r = MOD (x, y)	remainder: $x - y * [x/y]$
r = SIGN (x, y)	transfer of sign: $\lvert x \rvert$ sign y
r = DIM (x, y)	positive difference: $x - \min(x,y)$
r = MAX (x, y, ...)	maximum of x, y, ...
r = MIN (x, y, ...)	minimum of x, y, ...
r = SQRT (x)	square root: $\sqrt{x}$
r = EXP (x)	exponentiation: e^x
r = LOG (x)	natural logarithm: $\log_e x$
r = LOG10 (x)	common logarithm: $\log_{10} x$
r = SIN (x)	sine x
r = COS (x)	cosine x
r = TAN (x)	tangent x
r = ASIN (x)	arc sine x
r = ACOS (x)	arc cosine x
r = ATAN (x)	arc tangent x
r = ATAN2 (x, y)	arc tangent x/y
r = SINH (x)	hyperbolic sine x
r = COSH (x)	hyperbolic cosine x
r = TANH (x)	hyperbolic tangent x

Tensor Principal Values and Magnitude Functions	
r = PMAX (T_{11}, T_{22}, T_{33}, T_{12}, T_{23}, T_{31})	maximum principal values
r = PMIN (T_{11}, T_{22}, T_{33}, T_{12}, T_{23}, T_{31})	minimum principal values
r = PMAX2 (T_{11}, T_{22}, T_{12})	maximum principal values (2D)
r = PMIN2 (T_{11}, T_{22}, T_{12})	minimum principal values (2D)
r = TMAG (T_{11}, T_{22}, T_{33}, T_{12}, T_{23}, T_{31})	magnitude of the deviatoric part

IF Functions	
r = IFLZ (*cond*, *rtrue*, *rfalse*)	if *cond* < 0.0, *rtrue* else *rfalse*
r = IFEZ (*cond*, *rtrue*, *rfalse*)	if *cond* = 0.0, *rtrue* else *rfalse*
r = IFGZ (*cond*, *rtrue*, *rfalse*)	if *cond* > 0.0, *rtrue* else *rfalse*

Array ⇒ Global Variable Functions	
r = SUM (x)	sum of x over all nodes or elements
r = SMAX (x)	maximum of x over all nodes or elements
r = SMIN (x)	minimum of x over all nodes or elements

Envelope Functions	
r = ENVMAX (x)	maximum of x over all previous time steps
r = ENVMIN (x)	minimum of x over all previous time steps

C. Command Summary

Database Editing Commands (page 16)

TITLE

sets the title to be written to the output database.

Variable Selection Commands (page 17)

SAVE *variable*$_1$, *variable*$_2$, ... or *option*$_1$, *option*$_2$, ...

transfers variables from the input database to the output database.

DELETE *variable*$_1$, *variable*$_2$, ...

marks an assigned variable as a temporary variable that will not be written to the output database.

Time Step Selection Commands (page 19)

TMIN *tmin*

sets the minimum selected time to *tmin*.

TMAX *tmax*

sets the maximum selected time to *tmax*.

NINTV *nintv*

sets the number of selected time intervals to *nintv* (delta offset).

ZINTV *nintv*

sets the number of selected time intervals to *nintv* (zero offset).

DELTIME *delt*

sets the selected time interval to *delt*.

ALLTIMES

selects all time steps between *tmin* and *tmax*.

TIMES [ADD,] t_1, t_2, ...

selects times t_1, t_2, etc.

STEPS [ADD,] n_1, n_2, ...

selects time steps n_1, n_2, etc.

HISTORY ON or OFF

controls whether history time steps are included in the selected time steps.

Mesh Limiting Commands (page 22)

ZOOM *xmin*, *xmax*, *ymin*, *ymax*, *zmin*, *zmax*

sets the limits of the mesh to be written to the output database.

VISIBLE [ADD or DELETE,] $block\_id_1$, $block\_id_2$, ...

limits the element blocks to be written to the output database.

Element Block Selection Commands (page 23)

BLOCKS [ADD or DELETE,] $block\_id_1$, $block\_id_2$, ...

selects the element blocks which have defined values for all following equations.

MATERIAL [ADD or DELETE,] $block\_id_1$, $block\_id_2$, ...

is exactly equivalent to a BLOCKS command.

SHOW *command*

displays the settings of parameters relevant to the *command.*

LIST *option*

displays database information.

HELP *option*

displays information about the ALGEBRA program.

LOG

requests that the log file be saved when the program is exited.

END

ends the equation input and begins the equation evaluation.

QUIT

ends the equation input and exits the program immediately without writing an output database.

D. Sample Session

The following is an example session with ALGEBRA. Text following the ALGEBRA prompt (ALG>) is supplied by the user. The program response (if any) is shown directly below the equation or command. Comments on the example are in *italics*.

```
ALG> LIST VARS

Database:  UD:[APGILKE.EXODUS]TAPE11.EXO;1

SAMPLE DATABASE FOR ALGEBRA

Number of coordinates per node       =     2
Number of nodes                      =   644
Number of elements                   =   480
Number of element blocks             =     1

Number of node sets                  =     0
Number of side sets                  =     0

Code:  MISCPROG  version  1.0       on  12/23/85  at  10:21:59

ALG> LIST STEPS

Number of time steps = 21 (including 0 history-only)
   Minimum time =  0.00
   Maximum time = 10.00
```

```
ALG> SHOW TMAX
Select all times from 0.0 to 10.0
   Number of selected times = 21
ALG> TMAX 5.0
Select all times from 0.0 to 5.0
   Number of selected times = 11
ALG> NINTV 5
Select times 0.0 to 5.0 in 5 intervals with delta offset
   Number of selected times = 5
```

These commands select up to 5 time steps between 0.0 and 5.0 starting at an offset (1.0) from 0.0. The steps with the times nearest 1.0, 2.0, 3.0, 4.0, and 5.0 are selected. The equations are evaluated and the results written to the output database only for the selected steps.

```
ALG> LIST NAMES

Coordinate names: R Z

Variables Names:
   History:
   Global:   RESIDUAL  ENERGY   NORM     L2NORM
   Nodal:    DISPLR    DISPLZ   VELR     VELZ     ACCELR   ACCELZ
   Element:  SIGR      SIGZ     SIGT     TAURZ    EPSR     EPST
             EPSRZ
```

```
ALG> SAVE NODAL
```

*All the input database nodal variables (*DISPLR, DISPLZ, ..., ACCELZ*) will be written unchanged to the output database (unless they are assigned a value or listed in a* DELETE *command).*

```
ALG> VONMISES = (1.0/SQRT(2.0)) * TMAG(SIGR,SIGZ,SIGT,TAURZ,0,0)
ALG> EFFSTR = SQRT(1.5) * 5.79E-3 * VONMISES**4 * EXP(-12.0/300.0*1.987)
ALG> PRESS = (SIGR + SIGZ + SIGT) / 3.0
ALG> PRESS100 = (SIGR$100 + SIGZ$100 + SIGT$100) / 3.0
ALG> PHI = EFFSTR - 0.023 - PRESS * (4.43E-8 - 3.7E-15 * PRESS)
ALG> ALPHA = SIGR$56
ALG> BETA = ALPHA + 1.414
```

Assign element variables VONMISES, EFFSTR, PRESS, *and* PHI *and global variables* PRESS100, ALPHA, *and* BETA. *Note that the* PRESS100 *equation could be replaced by "*PRESS100 = PRESS$100*".*

```
ALG> DELETE ALPHA
```

ALPHA *(assigned in the equation "*ALPHA = SIGR$56*" above) becomes a temporary variable and will not be written to the output database.*

```
ALG> BAD = (A + 1)) + SIN (1,2)
*** Expected 1 parameter(s) for function SIN, found 2
*** Parenthesis do not balance
*** "A" is not a database variable
    Equation ignored
```

This equation contains several errors. Each error is flagged and the equation is ignored.

```
ALG> END
```

No further user input is accepted and the equation evaluation begins.

E. Site Supplements

E.1 VAX VMS

The command to execute ALGEBRA on VMS is:

ALGEBRA *input_database output_database user_input*

Input_database is the filename of the input EXODUS database. A prompt appears if *input_database* is omitted. The default is TAPE11.EXO.

Output_database is the filename of the output EXODUS database. A prompt appears if *output_database* is omitted. The default is TAPE12.EXO.

If *user_input* is given, the user input is read from this file. Otherwise user input is read from SYS$INPUT (the terminal keyboard). User output is directed to SYS$OUTPUT (the terminal).

ALGEBRA operates in either interactive or batch modes.

E.2 CRAY CTSS

To execute ALGEBRA, the user must have selected the acclib library and be running ccl.

The command to execute ALGEBRA on CTSS is:

algebra *input_database output_database* i=*input* o=*output*

Input_database is the filename of the input EXODUS database. The default is **tape11.**

Output_database is the filename of the output EXODUS database. The default is **tape11.**

User input is read from *input*, which defaults to **tty** (the terminal keyword). User output is directed to *output*, which defaults to **tty** (the terminal).

Distribution:

1510 J. W. Nunziato
1511 D. K. Gartling
1520 L. W. Davison
1521 R. D. Krieg and Staff (12)
1522 R. C. Reuter, Jr. and Staff (15)
1523 J. H. Biffle and Staff (12)
1523 A. P. Gilkey (30)
1524 A. K. Miller and Staff (12)
1530 W. Herrmann, Actg.
1531 S. L. Thompson
1533 S. T. Montgomery
1550 C. W. Peterson, Jr.
1556 W. L. Oberkampf
3141 S. A. Landenberger (5)
3151 W. I. Klein (3)
3154-1 for DOE/OSTI (8)
6258 D. S. Preece
6334 H. J. Iuzzolino
6334 R. D. McCurley
6334 J. S. Rath
6334 R. P. Rechard
6334 E. Shepherd
8240 C. W. Robinson
8241 G. A. Benedetti
8242 M. R. Birnbaum
8243 M. L. Callabresi
8244 C. M. Hartwig
8245 R. J. Kee
8524 P. W. Dean